Weekly Report(July.1.2019-July.7.2019)

Shanghai Jiao Tong University
1747836307@qq.com
Shaopu Rui

Abstract

This week I spend some time to build my environment on Linux and build a very simple demo for AlexNet.

1 Work Environment

Since the main memory of my computer is only 4G, it's really slow to run code on Linux, which is installed on HDD. So I used to run code in Windows. However, it's troublesome to configure some environment variables. So i decide to run code on Linux now. This week I just spend some time to install all the corresponding softwares and configur correctly. I also try to build a sample network from Alex Net. Although it wworks with many bugs.

1.1 NVIDIA

use order "lspci — grep -i nvidia" to see the GPU version.

1.2 Anaconda3

Download the Anaconda3 installation package which python version is 3.7 on the following website. Install it as requested.

`https://www.anaconda.com`

Since I did some modification on my terminal, I use zsh instead of bash, therefore I encountered a problem "zsh: command not found: conda". To solve it, I add "export PATH="/home/rsp/anaconda3/bin:$PATH" in the file loacted in the file path " /.zshrc", then type "source /.zshrc" to activate it.

1.3 virtualenv

Since we have installes Anaconda3, we can create an independent environment by the order "conda create -n env-name python=3.6", type "conda activate env-name" to get into the virtual environment, and focus on our work after installing some compulsory python libraries.

"conda deactivate" can get out of the virtual environment, and "conda remove -n env-name" can remove the virtual environment at all.

1.4 Pytorch

I encountered a big problem here, which cost me for a night and I still can't solve it. Finally I consult to my roomate Ni, he solved it for me. I really appreciate it.

The problem is I can't use gpu after installing cuda and cuDnn. At first I install latest version, it didn't works. So I removed them and installed version 9.0 instead, but it still didn't work. I checked

for several times but can't know why. Finally it's caused by the pytorch, which I installed long time ago, I even forgot it. Reinstall pytorch with a lower version, it works!

2 paragraph

I think image classification is a process that for a given dataset, we use classification methods to try to divide them into different parts. Accuracy is to measure a method is good or not. We want to get a classification result with a good accuracy, so we need to keep trying to find a optimal method.

About my plan, actually I didn't have a very concrete plan. In general, I will obey the requirements given in miniproject and finish them step by step. First is the AlexNet, I plan to use one more week working on it. Then the pre-trained model, I plan to working on it until the end of summer semester.

3 AlexNet Demo

```
# model = nn.Sequential(

#     nn.Conv2d(in_channels=3, out_channels=channel_1, kernel_size=(11, 11), stride=1, padding=(2, 2), bias=True),
#     nn.ReLU(),
# #     nn.BatchNorm2d(num_features=num_feature_1),
#     nn.MaxPool2d(kernel_size=(3, 3), stride=2),

#     nn.Conv2d(in_channels=channel_1, out_channels=channel_2, kernel_size=(5, 5), stride=1, padding=(2, 2), bias=T
#     nn.ReLU(),
# #     nn.BatchNorm2d(num_features=num_feature_2),
#     nn.MaxPool2d(kernel_size=(3, 3), stride=2),

#     nn.Conv2d(in_channels=channel_2, out_channels=channel_3, kernel_size=(3, 3), stride=1, padding=(1, 1), bias=T
#     nn.ReLU(),
# #     nn.BatchNorm2d(num_features=num_feature_3),

#     nn.Conv2d(in_channels=channel_3, out_channels=channel_4, kernel_size=(3, 3), stride=1, padding=(1, 1), bias=T
#     nn.ReLU(),
# #     nn.BatchNorm2d(num_features=num_feature_3),

#     nn.Conv2d(in_channels=channel_4, out_channels=channel_5, kernel_size=(3, 3), stride=1, padding=(1, 1), bias=T
#     nn.ReLU(),
# #     nn.BatchNorm2d(num_features=num_feature_2),
#     nn.MaxPool2d(kernel_size=(3, 3), stride=2),

#     Flatten(),

#     nn.Linear(in_features=feature_1, out_features=feature_2),
#     nn.ReLU(),
#     nn.Dropout(p=0.5, inplace=False),

#     nn.Linear(in_features=feature_2, out_features=feature_2),
#     nn.ReLU(),
#     nn.Dropout(p=0.5, inplace=False),

#     nn.Linear(in_features=feature_2, out_features=feature_3)
# )
```

Figure 1: AlexNet

```
tensor([8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8], device='cuda:0')
tensor([8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
        8, 8, 8, 8, 8, 8, 8, 8], device='cuda:0')
```

Figure 2: prediction output

I build a classic AlexNet like this, but a question puzzles me. The size of the images for training are all 3*32*32, but in the classic AlexNet, the size of input image are 3*227*227. I can't figure

out why they are different. And if it's the truth, how should I appply this dataset to train classic AlexNet? I know "torchvision.transforms" can resize the image, but I searched for this and didn't find someone else did such operation when training AlexNet. So does it necessary to resize it or use some other methds?

Another problem is I can't get a reasonable accuracy. I build the net and train it use the code above, but the outputs are always similar. Since the input is too big, I will show you the predict labels. They are always the same number, which varies from 0 to 9, corresponding to the number of classes. I thought the resize operation may cause this result, so I comment it and rebuild the net. The results are still the same. I think it may be caused by trainning process code, but I can't figure out why.

```python
def check_accuracy(loader, model):
    if loader.dataset.train:
        print('Checking accuracy on validation set')
    else:
        print('Checking accuracy on test set')

    num_samples = 0
    num_correct = 0
    model.eval()
    for x, y in loader:
        x = x.to(device=device)
        y = y.to(device=device)
        scores = model(x)
        _, preds = torch.max(scores, dim=1)
        print(preds)
        num_correct += (preds == y).sum()
        num_samples += x.shape[0]
    acc = float(num_correct) / num_samples
    print('Got %d / %d correct (%.2f%%)' % (num_correct, num_samples, 100 * acc))
    return acc
```

```python
def train(model, optimizer, batch_size, epochs=1):
    loader_train, loader_val = getData(batch_size)

    model = model.to(device=device)
    result = []
    for e in range(epochs):
        for t, (x, y) in enumerate(loader_train):
            x = x.to(device=device)
            y = y.to(device=device)
            optimizer.zero_grad() # 梯度归零
            model.train()
            scores = model(x)
            loss = F.cross_entropy(scores, y)  # 计算loss
            loss.backward() #  反向传播
            optimizer.step() # 更新参数
            print('Iteration %d, loss = %.4f' %(t, loss.item()))
        acc = check_accuracy(loader_val, model)
        result.append(acc)
        print()
```

Figure 3: main training process

4 Plan

I will appreciate it so much if someone can answer these questions for me so that I can start to solve next porblem. The sooner, the better.